

# Package ‘OrdinalCompositions’

June 19, 2026

**Type** Package

**Title** Wasserstein-Based Regression for Ordinal Compositional Data

**Version** 0.1.0

**Date** 2026-06-05

**Maintainer** Nicola Piras <nicola.piras97@unica.it>

**Description** Tools analyzing regression models for ordinal compositional data using Wasserstein-based distances. The package includes linear programming solvers under simplex constraints, tensor product constructions and performance metrics.

**Depends** R (>= 3.5.0)

**Imports** lpSolveAPI, stats, extraDistr

**Suggests** codalm, knitr, rmarkdown

**License** GPL-2

**NeedsCompilation** no

**Author** Nicola Piras [aut, cre],  
Monica Musio [aut],  
Beniamino Cappelletti-Montano [aut]

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**VignetteBuilder** knitr

**LazyData** true

**Repository** CRAN

**Date/Publication** 2026-06-19 15:50:18 UTC

## Contents

CitySatisf . . . . .	2
compute_R2 . . . . .	3
compute_wfrechet_mean . . . . .	4
default_lambda_grid . . . . .	5
educFM . . . . .	5

OCC . . . . .	6
opi . . . . .	7
ordinal_regression_simplex . . . . .	8
safe_dirichlet . . . . .	10
select_lambda . . . . .	11
solve_simplex_lp . . . . .	13
tensor_product . . . . .	14
wd . . . . .	14
wd_matrix . . . . .	15

<b>Index</b>	<b>17</b>
--------------	-----------

---

CitySatisf	<i>City Life Satisfaction Dataset</i>
------------	---------------------------------------

---

## Description

A dataset describing the satisfaction of residents with the quality of life in 83 European cities.

## Usage

CitySatisf

## Format

A data frame with 83 observations and 33 variables:

**City** city name  
**A\_15\_24, A\_25\_39, A\_40\_54, A\_55\_100** age classes  
**DH\_RS, DH\_RU, DH\_VS, DH\_VU** health care satisfaction  
**GS\_RS, GS\_RU, GS\_VS, GS\_VU** green spaces  
**QA\_RS, QA\_RU, QA\_VS, QA\_VU** air quality  
**SL\_RS, SL\_RU, SL\_VS, SL\_VU** life satisfaction  
**GJ\_RS, GJ\_RU, GJ\_VS, GJ\_VU** job opportunities  
**SN\_RS, SN\_RU, SN\_VS, SN\_VU** safety  
**GH\_RS, GH\_RU, GH\_VS, GH\_VU** housing

## Details

The data measures subjective well-being and satisfaction with urban living conditions across multiple domains such as healthcare, environment, safety, employment opportunities, and housing.

Responses are categorized into four levels: Very satisfied (VS), Rather satisfied (RS), Rather unsatisfied (RU), Very unsatisfied (VU).

The dataset can be used to study compositional ordinal regression models relating individual perceptions of urban quality of life to demographic groups.

The variable SL is typically used as a global indicator of life satisfaction.

**Source**

European urban quality-of-life survey.

---

compute\_R2

*Wasserstein R-squared*

---

**Description**

Computes an R-squared measure based on Wasserstein distances for compositional ordinal data.

**Usage**

```
compute_R2(Y, X, A, weights)
```

**Arguments**

Y	Response matrix
X	Input matrix
A	Transformation matrix
weights	Distance weights

**Details**

The Wasserstein coefficient of determination  $R_W^2$  is defined as the proportional reduction in error relative to the baseline:

$$R_W^2 = 1 - \frac{Dres}{Dtot}$$

where as baseline the Fréchet mean is used.

**Value**

A list with R2, Dres, Dtot, Fréchet mean and predictions

---

compute\_wfrechet\_mean *Compute the Wasserstein Fréchet mean of compositional data*

---

### Description

Computes the Fréchet mean in the Wasserstein geometry on the simplex. Given a sample of compositions, the cumulative distribution functions (CDFs) are first computed row-wise, the component-wise median of the CDFs is then obtained, and finally transformed back to the simplex by successive differencing.

### Usage

```
compute_wfrechet_mean(mat)
```

### Arguments

mat                    A numeric matrix whose rows are compositions.

### Details

Let  $P'_1, \dots, P'_N \in \Delta_m$  be the observed compositions and  $F_k(P'_i)$  the  $k$ -th component of their cumulative distribution functions. The Fréchet mean is defined through

$$\bar{F}_k = \text{median}\{F_k(P'_1), \dots, F_k(P'_N)\},$$

and the corresponding composition is recovered from the cumulative representation by finite differences.

### Value

A numeric vector representing the Wasserstein Fréchet mean of the sample compositions.

### Examples

```
Y <- rbind(
  c(0.2, 0.5, 0.3),
  c(0.1, 0.6, 0.3),
  c(0.4, 0.2, 0.4)
)

compute_wfrechet_mean(Y)
```

---

default\_lambda\_grid     *Lambda grid*

---

**Description**

Lambda grid

**Usage**

```
default_lambda_grid()
```

**Details**

Numeric vector of tested values for the regularization parameter  $\lambda$

**Value**

Numeric vector of lambda values

---

educFM     *Education level of father (F) and mother (M)*

---

**Description**

Education level of father (F) and mother (M) in percentages of low (l), medium (m), and high (h) educational attainment across European countries.

**Usage**

```
data(educFM)
```

**Format**

A data frame with 31 observations and 8 variables:

**country** Country identifier

**F.l** Percentage of females with low education level

**F.m** Percentage of females with medium education level

**F.h** Percentage of females with high education level

**M.l** Percentage of males with low education level

**M.m** Percentage of males with medium education level

**M.h** Percentage of males with high education level

**Details**

The dataset contains compositional information on education levels for fathers and mothers across 31 European countries.

This dataset is originally provided in the context of compositional data analysis and is also available in the package **robCompositions**.

The data are expressed in percentages and represent compositional parts of educational attainment distributions. They are commonly used in compositional data analysis and regression models on the simplex.

**Author(s)**

Peter Filzmoser, Matthias Templ

**Source**

Eurostat, <https://ec.europa.eu/eurostat/>

**References**

Filzmoser, P. & Templ, M. (various works on compositional data analysis)

---

OCC

*Ordinal Correlation Coefficient (OCC)*

---

**Description**

Computes the Spearman rank correlation between two sets of ordinal probability distributions using their centers of mass.

**Usage**

OCC(A, B)

**Arguments**

A                    Numeric matrix where each row represents a probability distribution  
B                    Numeric matrix with the same number of rows as A

**Details**

The center of mass of each distribution is computed as the expected value over the ordinal support. The Spearman correlation is then calculated between the resulting vectors.

**Value**

A numeric value representing the Spearman correlation coefficient

**See Also**[cor](#)**Examples**

```
A <- matrix(c(0.2,0.3,0.5,
              0.1,0.4,0.5), nrow = 2, byrow = TRUE)
B <- matrix(c(0.3,0.3,0.4,
              0.2,0.3,0.5), nrow = 2, byrow = TRUE)

OCC(A, B)
```

opi

*Ordinal Preservation Index***Description**

This index represents the proportion of data pairs whose ordinal rank is preserved by the model among all pairs with distinct observed responses.

**Usage**

```
opi(weights, P, Q, tol = 1e-08)
```

**Arguments**

weights	Numeric vector of weights
P	Numeric vector representing the first distribution
Q	Numeric vector representing the second distribution
tol	Numeric tolerance for comparison (default: 1e-8)

**Details**

The function compares distributions by evaluating their Wasserstein distances to each unit vector basis. It determines the proportion of pairs for which the ordering is preserved.

The index is defined as  $OPI = A/B$ , where  $A$  is the number of pairs  $(i, j)$  with  $i < j$  such that  $\alpha(P'_i, P'_j)\alpha(f(P_i), f(P_j)) = 1$ , and  $B$  is the total number of pairs with  $i < j$ .

The function  $\alpha(P, Q)$  is defined as:

- 1 if  $P \prec Q$
- -1 if  $P \succ Q$
- 0 otherwise

Here,  $\prec$  denotes the total Wasserstein order.

**Value**

- -1 if  $P < Q$
- 1 if  $P > Q$
- 0 if  $P$  and  $Q$  are equivalent

---

 ordinal\_regression\_simplex

*Ordinal Compositional Regression on the Simplex*


---

**Description**

Ordinal Compositional Regression on the Simplex

**Usage**

```
ordinal_regression_simplex(
  xdata,
  ydata,
  lambda1_grid = default_lambda_grid(),
  lambda2_grid = default_lambda_grid(),
  method = c("cv", "gcv"),
  K = NULL,
  weights = NULL,
  weights_product = NULL,
  return_tensor_index = FALSE,
  do_bootstrap = FALSE,
  B = 1000,
  compute_opi = FALSE,
  compute_R2 = FALSE,
  compute_OCC = FALSE,
  do_bootstrap_order = FALSE
)
```

**Arguments**

xdata	List of compositional predictors. <ul style="list-style-type: none"> <li>• SINGLE case: list of compositional vectors (each numeric vector)</li> <li>• TENSOR case: list of lists of compositional vectors</li> </ul>
ydata	List of compositional response vectors
lambda1_grid	Grid of tuning parameters for $\lambda_1$
lambda2_grid	Grid of tuning parameters for $\lambda_2$
method	Character string specifying the regularization selection criterion. Either "cv" (cross-validation) or "gcv" (generalized cross-validation).

K	Number of folds for cross-validation. If NULL, leave-one-out cross-validation is performed.
weights	Wasserstein weights (default uniform)
weights_product	Weights used in the construction of the tensor product (default uniform)
return_tensor_index	Logical, return tensor index mapping
do_bootstrap	Logical, bootstrap inference using Wasserstein distance between matrices
B	Number of bootstrap samples
compute_opi	Logical, OPI index
compute_R2	Logical, Wasserstein R2
compute_OCC	Logical, OCC index
do_bootstrap_order	Logical, ordering bootstrap

### Details

The function estimates a regression operator between compositional predictors and responses using Wasserstein geometry. Regularization is selected via generalized cross-validation. Optional bootstrap procedures provide confidence regions and additional statistics. The function supports both single compositional regression and multi-compositional tensor regression. In tensor mode, a design matrix is automatically constructed via tensor product:

$$Z_i = x_i^{(1)} \otimes x_i^{(2)} \otimes \dots$$

The user does NOT need to precompute tensor products.

### Value

A list containing:

- `A_hat`: Estimated regression matrix
- `bootstrap`: Bootstrap results (if requested). In the case of bootstrap procedure via Wasserstein distance between matrices the Fréchet mean is computed. In the case of bootstrap procedure via the Wasserstein total order the inf, sup and median for each column are computed.
- `OCC`: Ordinal Correlation Coefficient (if requested)
- `R2`: Wasserstein  $R^2$  Index (if requested)
- `OPI`: Ordinal Preservation Index (if requested)

### Examples

```
if (requireNamespace("codalm", quietly = TRUE)) {
  data(educFM)

  # --- preprocessing ---
```

```

father <- as.matrix(educFM[, 2:4])
ya <- father / rowSums(father)

mother <- as.matrix(educFM[, 5:7])
xa <- mother / rowSums(mother)

x <- cbind(xa[,3], xa[,2], xa[,1])
y <- cbind(ya[,3], ya[,2], ya[,1])

ydata <- split(y, seq_len(nrow(y)))
xdata <- split(x, seq_len(nrow(x)))

weights_orig <- rep(1, ncol(x) - 1)

# --- model ---
res <- ordinal_regression_simplex(
  xdata, ydata,
  weights = weights_orig, lambda2_grid=0, compute_opi=TRUE, compute_R2=TRUE, compute_OCC=TRUE
)
}

```

---

safe\_dirichlet

*Safe Dirichlet Sampling*


---

### Description

Generates a Dirichlet random vector ensuring numerical stability by enforcing a minimum threshold on parameters.

### Usage

```
safe_dirichlet(alpha, scale = 10, eps = 1e-06)
```

### Arguments

alpha	Numeric vector of Dirichlet parameters
scale	Scaling factor applied to alpha
eps	Minimum threshold to avoid zeros or non-finite values

### Value

A numeric vector sampled from a Dirichlet distribution

---

select_lambda	<i>Selection of Regularization Parameters</i>
---------------	---

---

**Description**

Selects the optimal pair of regularization parameters  $\lambda_1$  and  $\lambda_2$  for a Wasserstein-based model. The selection can be performed either through K-fold cross-validation (method = "cv") or generalized cross-validation (method = "gcv").

**Usage**

```
select_lambda(
  P,
  P_prime,
  weights,
  lambda1_grid,
  lambda2_grid,
  method = c("cv", "gcv"),
  K = NULL,
  verbose = FALSE
)
```

**Arguments**

P	List of input probability vectors (predictors).
P_prime	List of target probability vectors (responses).
weights	Numeric vector of weights used in the loss function.
lambda1_grid	Numeric vector of candidate values for the first regularization parameter.
lambda2_grid	Numeric vector of candidate values for the second regularization parameter.
method	Character string specifying the selection criterion. Either "cv" (cross-validation) or "gcv" (generalized cross-validation).
K	Number of folds for cross-validation. If NULL, leave-one-out cross-validation is performed.
verbose	Logical; if TRUE, progress information is printed.

**Details**

For each pair  $(\lambda_1, \lambda_2)$  in the supplied grids, the function estimates the transport matrix  $A$  by solving the constrained optimization problem implemented in [solve\\_simplex\\_lp](#).

When method = "cv", the optimal parameters are selected by minimizing the prediction error computed on validation folds. The loss is based on weighted Wasserstein-type discrepancies between cumulative distributions.

When method = "gcv", the fit is computed on the full sample and the effective degrees of freedom are estimated from the singular values of  $A$  according to

$$df = \sum_j \frac{\sigma_j^2}{\sigma_j^2 + \lambda_1 + \lambda_2},$$

where  $\sigma_j$  are the singular values of  $A$ . The GCV score is then given by

$$GCV(\lambda_1, \lambda_2) = \frac{\text{fit}}{(n - df)^2}.$$

### Value

A list containing the selected regularization parameters and the corresponding validation scores.

If method = "cv", the returned list contains:

- method: "cv".
- best\_lambda1: selected value of  $\lambda_1$ .
- best\_lambda2: selected value of  $\lambda_2$ .
- cv\_values: matrix of cross-validation errors.
- lambda1\_grid: grid of candidate  $\lambda_1$  values.
- lambda2\_grid: grid of candidate  $\lambda_2$  values.
- K: number of folds.

If method = "gcv", the returned list contains:

- method: "gcv".
- best\_lambda1: selected value of  $\lambda_1$ .
- best\_lambda2: selected value of  $\lambda_2$ .
- score\_values: GCV scores for all parameter combinations.
- fit\_values: fit values for all parameter combinations.
- lambda1\_grid: grid of candidate  $\lambda_1$  values.
- lambda2\_grid: grid of candidate  $\lambda_2$  values.

### See Also

[solve\\_simplex\\_lp](#)

### Examples

```
# Example with simple synthetic data
P <- list(c(0.5, 0.5), c(0.3, 0.7))
P_prime <- list(c(0.4, 0.6), c(0.2, 0.8))
w <- c(1)
lambda_grid <- c(0, 0.01, 0.1)

select_lambda(P, P_prime, w, lambda_grid, lambda_grid)
```

---

solve\_simplex\_lp      *Solve Simplex-Constrained Linear Program*

---

### Description

Solves a linear programming problem with simplex constraints and optional regularization for Wasserstein-based fitting.

### Usage

```
solve_simplex_lp(P, P_prime, weights, lambda1 = 0, lambda2 = 0)
```

### Arguments

P	List of input probability vectors
P_prime	List of target probability vectors
weights	Numeric vector of weights
lambda1	First-order Regularization parameter
lambda2	Second-order Regularization parameter

### Details

Consider a sample of  $N$  pairs  $\{(P_i, P'_i)\}_{i=1}^N$ . Here,  $P_i \in \Delta_n$  denotes the predictor, while  $P'_i \in \Delta_m$  denotes the ordinal response distribution. We assume a linear dependence structure defined by:

$$f(P) = AP$$

where the regression coefficient  $A \in CS^{(m+1) \times (n+1)}$  is a column-stochastic (CS) matrix.

The estimation of  $A$  is performed by minimizing the loss function  $L(A)$ , defined as

$$L(A) = \sum_{i=1}^N d_a(AP_i, P'_i) = \sum_{i=1}^N \sum_{k=1}^m a_k |F_k(AP_i) - F_k(P'_i)|.$$

Once the associated cumulative representation  $F_{k,j} = \sum_{r=1}^k A_{rj}$  is define, the regularized objective function is:

$$\mathcal{L}_{reg}(A) = \min_{A \in CS} \sum_{i=1}^N d_a(AP_i, P'_i) + \lambda_1 \sum_{j=1}^n \sum_{k=1}^m a_k |F_{k,j+1} - F_{k,j}| + \lambda_2 \sum_{j=2}^n \sum_{k=1}^m a_k |F_{k,j-1} - 2F_{k,j} + F_{k,j+1}|$$

where  $\lambda_1, \lambda_2 \geq 0$  are tuning parameters controlling the trade-off between goodness-of-fit and the smoothness of the estimated conditional distributions.

### Value

A list containing the optimal matrix  $A$

---

tensor_product	<i>Tensor Product of Compositions</i>
----------------	---------------------------------------

---

**Description**

Computes the tensor product of multiple compositional vectors, optionally using weighted cumulative costs for ordering.

**Usage**

```
tensor_product(comps, weights = NULL, return_indices = FALSE)
```

**Arguments**

comps	List of compositional vectors
weights	Optional list of weight vectors
return_indices	Logical; whether to return index combinations

**Value**

A list containing the product vector (and optionally indices)

**Examples**

```
P <- c(0.2, 0.3, 0.5)
Q <- c(0.4, 0.3, 0.2, 0.1)
a <- c(5, 15)
b <- c(2, 4, 8)
tensor_product(comps=list(P,Q), weights = list(a,b), return_indices = TRUE)
```

---

wd	<i>Weighted Wasserstein Distance (W1)</i>
----	---

---

**Description**

Computes the weighted first-order Wasserstein distance between two discrete probability distributions defined on an ordered support.

**Usage**

```
wd(weights, P, Q)
```

**Arguments**

weights	Numeric vector of non-negative weights of length n
P	Numeric vector representing the first probability distribution in the Simplex $\Delta_n$
Q	Numeric vector representing the second probability distribution in the Simplex $\Delta_n$

**Details**

The distance is computed as the weighted sum of absolute differences between the cumulative distribution functions of P and Q, excluding the last component.

$$d_a(p, q) = \sum_{k=1}^n a_k |F_p(k) - F_q(k)|$$

where  $F_p(k) = \sum_{j=1}^k p_j$  and  $F_q(k)$  is defined analogously.

**Value**

A numeric value representing the weighted Wasserstein distance

---

wd_matrix	<i>Matrix Wasserstein Distance</i>
-----------	------------------------------------

---

**Description**

Computes the weighted Wasserstein distance between two matrices by summing the column-wise Wasserstein distances.

**Usage**

```
wd_matrix(A, B, weights)
```

**Arguments**

A	Numeric matrix where each column represents a probability distribution. $A \in CS^{(m+1) \times (n+1)}$ where a matrix $A \in \mathbb{R}^{(m+1) \times (n+1)}$ is said to be <i>column-stochastic</i> (CS) if its entries $A_{ij} \geq 0$ for all $i, j$ , and $\sum_{i=1}^{m+1} A_{ij} = 1$ for all $j = 1, \dots, n + 1$ .
B	Numeric matrix with the same dimensions as A
weights	Numeric vector of weights of length m

**Details**

The distance is computed column by column using cumulative distributions and then aggregated across all columns.

$$D_a(A, B) := \sum_{j=1}^{n+1} d_a(A_{.j}, B_{.j}) = \sum_{j=1}^{n+1} \sum_{k=1}^m a_k |F_k(A_{.j}) - F_k(B_{.j})|.$$

**Value**

A numeric value representing the total Wasserstein distance

# Index

## \* datasets

CitySatisf, 2  
educFM, 5

CitySatisf, 2  
compute\_R2, 3  
compute\_wfrechet\_mean, 4  
cor, 7

default\_lambda\_grid, 5

educFM, 5

OCC, 6  
opi, 7  
ordinal\_regression\_simplex, 8

safe\_dirichlet, 10  
select\_lambda, 11  
solve\_simplex\_lp, 11, 12, 13

tensor\_product, 14

wd, 14  
wd\_matrix, 15